UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/552,643	10/07/2005	Steve D. Taylor	109874-145266	2626

25943 7590 11/12/2008

SCHWABE, WILLIAMSON & WYATT, P.C.
PACWEST CENTER, SUITE 1900
1211 SW FIFTH AVENUE
PORTLAND, OR 97204

EXAMINER	
BROPHY, MATTHEW J	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
11/12/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/552,643	TAYLOR, STEVE D.	
	Examiner	Art Unit	
	MATTHEW J. BROPHY	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 August 2008*.

2a)☒ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,10-22 and 28-41* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4,10-22 and 28-41* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _____ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *8/25/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed August 25, 2008.

2. Claims 1-4, 10-22 and 28-41 are pending; 23-27 are canceled, 33-41 are new.

3. 35 U.S.C. §101 and §112 rejections are withdrawn in view of applicants amendment.

Claim Rejections - 35 USC § 102

> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

1. Claims 1-4 and 10-22, 28-41 are rejected under 35 U.S.C. 102(b) as being anticipated by 5,850,548 Williams et al hereinafter Williams.

2. The text of claim amendments not specifically address here are rejected based on the same basis that the respective claims were in the previous office action.

Regarding Claim 1, Williams teaches: a method comprising: a first display container cell **(e.g. Column 9, Lines 14-26, "As illustrated in FIG. 5A, the Visual Development System 230 of the present invention includes a development interface or work surface 500. The work surface 500, in turn, includes a main window 501, a Component Inspector Window 520, a Component Manager Window 530, and a Library Window 540. The main window 501 includes a menu bar 505 and a tool bar 510. Menu bar 505 includes user-selectable menu choices for invoking operations of the system. Tool bar 510 includes an array of screen buttons for one-click access to system commands. Other windows on the work surface 500 (e.g., window 540) may register their own tools with the tool bar or palette 510, for providing users with access to those tools from the main window 501.")**; a second

display container cell nested within the first display container cell **(e.g. Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1.")**; and a first action cell nested within one of said first and second display container cells, the first action cell being associated with causing at least a first action to be performed in association with or on behalf of at least one of a third display container cell and a fourth display container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")** and the first action being conditioned at least in part on one or more state values or policy attributes

of one or more of the display container cells **(Col 11, Ln 41-53, "As shown in FIG. 6E, for instance, by selecting VisualComponent3 625 and requesting property inspection, such as "right-clicking" it, the system displays pop-up menu 626. As shown, the pop-up menu 626 includes a "Show Property" selection which brings up a sub-menu having a "Value" choice 627 (which corresponds to Value property 655). As shown in FIG. 6F, VisualComponent3 625 includes Value box 628--a visual representation of the "Value" property of VisualComponent3. At this instance in the example, the Value property of VisualComponent3 is not "connected" to anything; it is essentially "free floating." Next, the steps for connecting the Value property will be described in detail.")**

Regarding Claim 2, Williams teaches: wherein the first display container cell is a selected one of a root region container cell and a nested region container cell **(e.g. Column 11, Lines 16-28, "The example will continue with an illustration of how the user connects one component to another, such as connecting the Scroll bar 617 to the Edit Field (so that movement of the Slider causes information to appear in the Edit Field). As VisualComponent2 620 and VisualComponent3 625 appear in VisualComponent1 630, as shown in FIG. 6C, it is not evident exactly what is inside either one of the two components (i.e., the VisualComponent2 and the VisualComponent3 nested components). In other words, at this point the system has encapsulated the complexity of the nested components so that they do not manifest properties at this particular instance.")**.

Regarding Claim 3, Williams teaches: wherein the second display container cell is a display container cell selected from a display container cell group consisting of a nested region container cell, a nested zone container cell, and a nested action cell pool container cell **(e.g. Column 11, Lines 16-28, "The example will continue with an illustration of how the user connects one component to another, such as connecting the Scroll bar 617 to the Edit Field (so that movement of the Slider causes information to appear in the Edit Field). As VisualComponent2 620 and VisualComponent3 625 appear in VisualComponent1 630, as shown in FIG. 6C, it is not evident exactly what is inside either one of the two components (i.e., the VisualComponent2 and the VisualComponent3 nested components). In other words, at this point the system has encapsulated the complexity of the nested components so that they do not manifest properties at this particular instance.")**.

Regarding Claim 4, Williams teaches: wherein the fourth display container cell is nested in the third display container cell **(e.g. Column 11, Lines 16-28, "The example will continue with an illustration of how the user connects one component to another, such as connecting the Scroll bar 617 to the Edit Field (so that movement of the Slider causes information to appear in the Edit Field). As VisualComponent2 620 and VisualComponent3 625 appear in VisualComponent1 630, as shown in FIG. 6C, it is not evident exactly what is inside either one of the two components (i.e., the VisualComponent2 and the VisualComponent3 nested components). In other words, at this point the system has encapsulated the complexity of the nested components so that they do not manifest properties at**

this particular instance." While this example of teaches 3 VisualComponents, it is inherent here that the could include several additional components, with multiple nesting of components).

Regarding Claim 10, Williams teaches: wherein the fourth display container cell is an invisible cell **(e.g. Column 16, Lines 15-29, "The Edit class implements the IControlServer methods 1020: ClientSet, ParentWindow, CreatedSet, VisibleSet, DesigningSet, DimensionSet, DimensionGet, and StateGet. These methods provide functionality typically associated with a screen control such as an Edit field. Their names describe their functionality. For instance, DimensionSet sets the dimensions for an Edit control, based on passed-in Values for starting Location (x, y), Width (dx) and Height (dy). Similarly, VisibleSet specifies whether the control is currently visible in the user interface (at runtime). This is the interface through which the parent window (i.e., container) interacts with the control, for example, creating it, turning its visibility on or off, sets its dimensions, or the like. When the user re-sizes the Edit control during design, the parent window in turn makes one or more DimensionSet calls.")**.

Regarding Claim 11, Williams teaches: wherein the first action cell is associated with causing a first action to be performed in association with or on behalf of both the third container cell and the fourth container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g.,**

by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.").

Regarding Claim 12, Williams teaches: wherein the first action cell is further associated with causing a second action to be performed in association with or on behalf of at least one of the third and the fourth container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding Claim 13, Williams teaches: wherein the first action cell is further associated with causing a second action to be performed in association with or on behalf of at least one of the third and the fourth container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal**

properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.").

Regarding Claim 14, Williams teaches: wherein the first action comprises performing an action selected from an action group consisting of adding a display container cell as a currently visible display contain cell among one or more other currently visible display container cell **(e.g. Column 9, Line 59 to Column 10, Line 10, "Throughout the user session, the work surface 500 is always "live." For instance, the Component Inspector Window 520 is updated to display specific information about the currently selected component (i.e., Window Component 551). In particular, the Component Inspector displays information about the component's Name, Caption, Position, Height, and the like. The user can easily modify a property by clicking on the desired property (e.g., "Caption" field) and entering in a new Value. Since the work space is always live, the component whose property has changed is immediately updated to reflect the newly entered Value. Any modifications made directly to the component (e.g., resizing window 560) are, in turn, immediately reflected by the Component Inspector 520; also**

shown, the Component Manager Window 530 is updated to display the new component, shown at 533. Since the system is always live, there is no "compile/link" cycle. Instead, properties and methods of components remain active at all times so that changes occurring in components take effect immediately. ") rendering a content, process a content (**e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")** and causing a script to be executed **(e.g. Column 13, Lines 18-42, "Referring now to FIGS. 9A-B, a Program Viewer of the present invention will be described. Super components are the system's controllers. They support program viewers, such as the program viewer 900 shown in FIG. 9A, as the main viewers of the system. The Program Viewer 900 provides facilities for editing programs, either in script or graphical (visual) programming format. The Viewer 900 has two principal regions: a structure pane 910 and a programming pane 920. All program viewers share the structure pane, which provides a representation of the structure of the project being edited. The**

programming pane, on the other hand, contains the program itself, either in script (shown at 925) or visual programming (shown at 927) representation.").

Regarding Claim 15, Williams teaches: wherein the first action comprises performing an action selected from an action group consisting of requesting a user action, requesting a local system action, and/or requesting a remote system action **(e.g. column 4, Lines 40-45, "System 200 includes a user interface (UI) 240, preferably a graphical user interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system 100 in accordance with instructions from operating module 210, Windows 215, and/or application modules 220, 225. The UI 240 also serves to display the results of an operation, whereupon the user may supply additional inputs or terminate the session" or e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component." Or Column 13, Lines 12-17, "The system allows multiple viewers to be opened on each component, with the system maintaining synchronization between the**

component and the various views. This is helpful in a multi-client environment, so that the program may be simultaneously edited from various machines. At all times, the end result of the work in progress appears on the screen of all clients.").

Regarding Claim 16, Williams teaches: wherein said rendering of the first action cell is performed in view of a plurality of associated attributes of the first action cell defining the first action cell, the associated attributes, the first action cell, including attributes selected from a attribute group consisting of one or more structural attributes, one or more geometric attributes, one or more visual attributes, one or more policy attributes, and one or more behavior attributes **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding Claim 17, Williams teaches: further comprising performing by a computing device the first action in response to a user selection of the first action cell, and the action being an action select from an action group consisting of associating a

defining attribute with a display container cell, modifying a defining attribute associated with a display container cell, and deleting a defining attribute associated with a display container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding Claim 18, Williams teaches: w further comprising performing by a computing device the first action in response to a user selection of the first action cell, and the action being an action select from an action selected from an action group consisting of adding one or more cells to the EUI, and deleting one or more cells from the EUI **(e.g. Column 9, Line 59 to Column 10, Line 10, "Throughout the user session, the work surface 500 is always "live." For instance, the Component Inspector Window 520 is updated to display specific information about the currently selected component (i.e., Window Component 551). In particular, the Component Inspector displays information about the component's Name, Caption, Position, Height, and the like. The user can easily modify a property by**

clicking on the desired property (e.g., "Caption" field) and entering in a new Value. Since the work space is always live, the component whose property has changed is immediately updated to reflect the newly entered Value. Any modifications made directly to the component (e.g., resizing window 560) are, in turn, immediately reflected by the Component Inspector 520; also shown, the Component Manager Window 530 is updated to display the new component, shown at 533. Since the system is always live, there is no "compile/link" cycle. Instead, properties and methods of components remain active at all times so that changes occurring in components take effect immediately. ").

Regarding Claim 19, Williams teaches: wherein the first action is conditioned at least in part on one or more attributes of one or more of the display container cells within which the first action cell is nested **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding Claim 20, Williams teaches: wherein the first action is conditioned at least in part on one or more state values or policy attributes and the state values and policy attributes are associated with a display container cell that is separate from the first, second third and fourth display container cells.**(e.g. Column 7, Lines 29-42, "Some components have more than one type of behavior and may reside in more than one hierarchy of objects. Conventional graphical user interface elements, such as screen buttons and the like, appear as simple, static rectangles in a "Logic View" of the program. In a "User Interface View," on the other hand, objects are shown having a more complex appearance (e.g., a push button or a spread sheet). The hierarchical location of controls in a window (i.e., inside child windows and group boxes) may have little or no bearing on their location in the Logic View. State diagrams may be constructed where the states and events correspond to components in the Logic View. The hierarchical structure of events and states in a state diagram do not necessarily appear in the same way in the Logic View.")** see also **(Col 11, Ln 41-53, "As shown in FIG. 6E, for instance, by selecting VisualComponent3 625 and requesting property inspection, such as "right-clicking" it, the system displays pop-up menu 626. As shown, the pop-up menu 626 includes a "Show Property" selection which brings up a sub-menu having a "Value" choice 627 (which corresponds to Value property 655). As shown in FIG. 6F, VisualComponent3 625 includes Value box 628--a visual representation of the "Value" property of VisualComponent3. At this instance in the example, the Value property of VisualComponent3 is not "connected" to anything; it is essentially**

"free floating." Next, the steps for connecting the Value property will be described in detail.")

Regarding Claim 21, Williams teaches: wherein the first action associated with the first action cell is an action to be performed in response to a structural exception event involving the at least one of the fourth and fifth display container cells **(Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1. In a manner similar to before, the user can then proceed to create Edit Field 615, by selecting Edit Component 611 from the Library Palette. The act of creating Edit Field 615 causes the Edit Field to appear within the window 610. Thus, VisualComponent2 is a "sub-component" of VisualComponent1--that is, it is a component nested within VisualComponent1. Moreover, the act of creating VisualComponent2 within the VisualComponent1 window causes it (i.e., the Edit Field) to represent itself within the Window object 610, which is the visual component corresponding to Windowl**

601. Thus as shown, Edit Field 615 appears within the boundaries of Window 610."0.

Regarding Claim 22, Williams teaches: wherein the EUI further comprises a second action cell associated with causing at least a second action to be performed in association with or on behalf of one or more nested container cells **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding Claim 28, Williams teaches: A system comprising: a processor **(Column 4, Lines 25-26, "The present invention may be embodied on a computer system such as the system 100 of FIG. 1, which includes a central processor 101,")**; a display coupled to the processor **(Column 4, Lines 30, "a display device 106")**; and storage coupled to the processor **(Column 4, Lines 31-32, "and a non-volatile or mass storage 107 (e.g., hard or fixed disk, optical disk, magneto-optical disk, or flash memory)."**, and having stored therein instructions designed implement an end user interface for the system, the end user interface having a first display container cell

(e.g. Column 9, Lines 14-26, "As illustrated in FIG. 5A, the Visual Development System 230 of the present invention includes a development interface or work surface 500. The work surface 500, in turn, includes a main window 501, a Component Inspector Window 520, a Component Manager Window 530, and a Library Window 540. The main window 501 includes a menu bar 505 and a tool bar 510. Menu bar 505 includes user-selectable menu choices for invoking operations of the system. Tool bar 510 includes an array of screen buttons for one-click access to system commands. Other windows on the work surface 500 (e.g., window 540) may register their own tools with the tool bar or palette 510, for providing users with access to those tools from the main window 501."); a second display container cell nested within the first display container cell **(e.g. Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1.")**; and a first action cell nested within one of said first and second display container cells, the first action cell being associated with causing at least a first action to be performed in association with or on behalf of at least one of a third container cell and a fourth container cell **(e.g.**

Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.") and the first action being conditioned at least in part on one or more state values or policy attributes of one or more of the display container cells **(Col 11, Ln 41-53, "As shown in FIG. 6E, for instance, by selecting VisualComponent3 625 and requesting property inspection, such as "right-clicking" it, the system displays pop-up menu 626. As shown, the pop-up menu 626 includes a "Show Property" selection which brings up a sub-menu having a "Value" choice 627 (which corresponds to Value property 655). As shown in FIG. 6F, VisualComponent3 625 includes Value box 628--a visual representation of the "Value" property of VisualComponent3. At this instance in the example, the Value property of VisualComponent3 is not "connected" to anything; it is essentially "free floating." Next, the steps for connecting the Value property will be described in detail.")**

Regarding Claim 29, Williams teaches:, wherein the method further comprises performing the action in response to a user selection of the first action cell, and the action being an action selected from an action group consisting of adding a display container cell as a currently visible display contain cell among one or more other currently visible display container cell, rendering a content, process a content and causing a script to be executed **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**..

Regarding Claim 30, Williams teaches: wherein the method further comprises performing the action in response to a user selection of the action cell, and the action being an action selected from an action group consisting of associating one or more defining attributes with a display container cell, modifying one or more defining attributes associated with a display container cell, deleting one or more defining attributes associated with a display container cell, adding one or more cells to the EUI, and deleting one or more cells from the EUI **(e.g. Column 11, Lines 29-40, "The user**

accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.").

Regarding Claim 31, Williams teaches:wherein the first action is conditioned at least in part on the one or more state values or policy attributes and the state values and policy attributes are associated with a display container cell that is separate and distinct from the first, second third and fourth display container cells. **(Col 11, Ln 41-53, "As shown in FIG. 6E, for instance, by selecting VisualComponent3 625 and requesting property inspection, such as "right-clicking" it, the system displays pop-up menu 626. As shown, the pop-up menu 626 includes a "Show Property" selection which brings up a sub-menu having a "Value" choice 627 (which corresponds to Value property 655). As shown in FIG. 6F, VisualComponent3 625 includes Value box 628--a visual representation of the "Value" property of VisualComponent3. At this instance in the example, the Value property of VisualComponent3 is not "connected" to anything; it is essentially "free floating." Next, the steps for connecting the Value property will be described in detail.")**

Regarding Claim 32, Williams teaches: wherein the EUI further comprises a second action cell associated with causing at least a second action to be performed in

association with or on behalf of one or more nested container cells **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.

Regarding the new claims Willams teaches:

Regarding Claim 33.(New) Williams teaches: In a computing environment, a method of operation comprising: creating by a computing device, for an end user interface, a first display container cell having a first content visibility **(e.g. Column 9, Lines 14-26, "As illustrated in FIG. 5A, the Visual Development System 230 of the present invention includes a development interface or work surface 500. The work surface 500, in turn, includes a main window 501, a Component Inspector Window 520, a Component Manager Window 530, and a Library Window 540. The main window 501 includes a menu bar 505 and a tool bar 510. Menu bar 505 includes user-selectable menu choices for invoking operations of the system. Tool bar 510 includes an array of screen buttons for one-click access to system commands. Other windows on the work surface 500 (e.g., window 540) may register their own**

tools with the tool bar or palette 510, for providing users with access to those tools from the main window 501.");

creating by a computing device, for the end user interface, a second display container cell a second content visibility **(e.g. Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1.")**;

receiving by a computing device, a first request to add a third display container cell to the end user interface, the third display container cell having a third content visibility **(Col 6, Ln 48-63 "Referring now to FIGS. 4A-G the overall process will be described. As shown in FIG. 4A, the user first creates a Blank Composite Component 400. The user assigns the component a Name 401, such as "My Program"; the Name serves as the name for the entire program.**

The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components. As shown in FIG. 4B, for instance, upon the user double-clicking the Component 400, the following

sub-components are displayed: In Calc sub-component 410, Processing sub-component 430, and Out Calc sub-component 420. These sub-components correspond to the main functional areas of this sample program. The user may then proceed to choose one of the newly created sub-components (e.g., Processing). The user can then "jump into" the sub-component by double-clicking on it.");

adding by a computing device, the third display container cell to the end user interface, including at least one of shifting at least one of the first or second display container cell to coalesce available space, downsizing at least one of the first or the second display container cell to increase available space, or removing at least one of the first or the second display container cell to increase available space,

(Col 6, Ln 52-55, "The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components", see e.g. 4A-G) *[here, the high-level component is removed (e.g. "My Program" Fig. 4E) from the display when the "jump" (e.g. to FIG. 4F) is completed]*

while maintaining said first or second content visibility when performing said at least one of shifting, downsizing or removing, if the corresponding first or second display container cell is not removed.

(Col 6, Ln 52-55, "The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components" see e.g. 4A-G) *[conversely, the sub-component (e.g. "processing" FIG. 4E) is visually maintained when shifting ("jump-inside") occurs (e.g. "processing FIG. 4F)]*

Regarding Claims 34, 37 and 40 (New) Williams teaches: wherein said first and second display container cell further having a first and a second kernel size respectively, and said first or second display container cell is removed if said first or second display container cell has to be downsized to smaller than the corresponding first or second kernel size. **(e.g. Col 16, Ln 15-30"The Edit class implements the IControlServer methods 1020: ClientSet, ParentWindow, CreatedSet, VisibleSet, DesigningSet, DimensionSet, DimensionGet, and StateGet. These methods provide functionality typically associated with a screen control such as an Edit field. Their names describe their functionality. For instance, DimensionSet sets the dimensions for an Edit control, based on passed-in Values for starting Location (x, y), Width (dx) and Height (dy). Similarly, VisibleSet specifies whether the control is currently visible in the user interface (at runtime). This is the interface through which the parent window (i.e., container) interacts with the control, for example, creating it, turning its visibility on or off, sets its dimensions, or the like. When the user re-sizes the Edit control during design, the parent window in turn makes one or more DimensionSet calls.")**

35.

Regarding Claims 35, 38 and 41 (New) Williams teaches: creating by a computing device, a display action cell nested in a selected

one of said first or second display container cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**.;.

receiving by a computing device, a user interaction with the display action cell **(e.g. Column 11, Lines 29-40, "The user accesses the VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.")**; and

in response, executing by a computing device, a binary associated with

the display action cell **(e.g. Column 11, Lines 29-40, "The user accesses the**

VisualComponent3 (i.e., access to its internal properties) by adding a new property to the component, as shown in FIG. 6C. The user requests property inspection of the Slider 651 which appears in the visual editor 650 (e.g., by right-clicking on it with a mouse cursor). In response, the system displays pop-up menu 652. The user may instruct the system to create a new property by selecting "New Property" menu choice 653. The system, in response, displays a Property Object 654, as shown in FIG. 6D. Here, the user can type in a New Property, such as "Value" 655. Now, Value 655 is a property which may be accessed from outside the component.").

Regarding Claim 36, Williams teaches: In a computing environment, a method of operation comprising:

creating by a computing device, for an end user interface, a first display container cell having a first content visibility; **(e.g. Column 9, Lines 14-26, "As illustrated in FIG. 5A, the Visual Development System 230 of the present invention includes a development interface or work surface 500. The work surface 500, in turn, includes a main window 501, a Component Inspector Window 520, a Component Manager Window 530, and a Library Window 540. The main window 501 includes a menu bar 505 and a tool bar 510. Menu bar 505 includes user-selectable menu choices for invoking operations of the system. Tool bar 510 includes an array of screen buttons for one-click access to system commands. Other windows on the work surface 500 (e.g., window 540) may register their own tools with the tool bar or palette 510, for providing users with access to those**

tools from the main window 501.")

creating by a computing device, for the end user interface, a second display container cell with a second content visibility **(e.g. Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1.")**;

receiving by a computing device, a request to expand a selected one of the first or second display container cell;

(Col 6, Ln 48-63 "Referring now to FIGS. 4A-G the overall process will be described. As shown in FIG. 4A, the user first creates a Blank Composite Component 400. The user assigns the component a Name 401, such as "My Program"; the Name serves as the name for the entire program.

The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components. As shown in FIG. 4B, for instance, upon the user double-clicking the Component 400, the following sub-components are displayed: In Calc sub-component 410, Processing sub-

component 430, and Out Calc sub-component 420. These sub-components correspond to the main functional areas of this sample program. The user may then proceed to choose one of the newly created sub-components (e.g., Processing). The user can then "jump into" the sub-component by double-clicking on it.")

expanding by a computing device, the selected one of the first or second display container cells, including one or more of

shifting the other one of the first or second display container cell to coalesce available space,

downsizing the other one of the first or second display container cell to increase available space, or

removing at least one of the first or the second display container cell to increase available space,

(Col 6, Ln 52-55, "The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components" see e.g. 4A-G)

while maintaining said first or second content visibility when performing said at least one of shifting, downsizing or removing, if the corresponding first or second display container cell is not removed. **(Col 6, Ln 52-55, "The user then proceeds to select (e.g., double-click) on the Component 400 to "jump inside" the component for creating sub-components" see e.g. 4A-G)**

Regarding Claim 39, Williams teaches:

creating by a computing device, for an end user interface, a first display container cell having a first content visibility; **(e.g. Column 9, Lines 14-26, "As illustrated in FIG. 5A, the Visual Development System 230 of the present invention includes a development interface or work surface 500. The work surface 500, in turn, includes a main window 501, a Component Inspector Window 520, a Component Manager Window 530, and a Library Window 540. The main window 501 includes a menu bar 505 and a tool bar 510. Menu bar 505 includes user-selectable menu choices for invoking operations of the system. Tool bar 510 includes an array of screen buttons for one-click access to system commands. Other windows on the work surface 500 (e.g., window 540) may register their own tools with the tool bar or palette 510, for providing users with access to those tools from the main window 501.")**

creating by a computing device, for the end user interface, a second display container cell with a second content visibility **(e.g. Column 10, Lines 26-49, "Components are nestable to an arbitrary depth within the visual programming editor. Any component which contains other components is known as a "SuperComponent"; a component which is contained is a "sub-component." Nesting is illustrated in FIG. 6A. As previously described, Window1 (shown at 601) is manifested by Window object 610, at the level of the user interface. An additional element, such as an Edit box or Scroll bar, can be added to the window 610. For instance, a new**

component--VisualComponent2 shown at 620--may be placed within the first Visual Component, VisualComponent1.");

receiving by a computing device, a request to contract a selected one of the first and second display container cells **(e.g. Col 16, Ln 15-30"The Edit class implements the IControlServer methods 1020: ClientSet, ParentWindow, CreatedSet, VisibleSet, DesigningSet, DimensionSet, DimensionGet, and StateGet. These methods provide functionality typically associated with a screen control such as an Edit field. Their names describe their functionality. For instance, DimensionSet sets the dimensions for an Edit control, based on passed-in Values for starting Location (x, y), Width (dx) and Height (dy). Similarly, VisibleSet specifies whether the control is currently visible in the user interface (at runtime). This is the interface through which the parent window (i.e., container) interacts with the control, for example, creating it, turning its visibility on or off, sets its dimensions, or the like. When the user re-sizes the Edit control during design, the parent window in turn makes one or more DimensionSet calls.")**;

contracting by a computing device, the selected one of the first and second display container cells, including one or more of shifting the other one of the first and second display container cells to reposition the other one of the first and second display container cells in view of newly available space resulted from the requested contracting, upsizing the other one of the first and second display container cells to consume all or some of the newly available space resulted from the requested contacting, or removing

the selected one of the first or second display container cell to be contracted, **(e.g. Col 16, Ln 15-30"The Edit class implements the IControlServer methods 1020: ClientSet, ParentWindow, CreatedSet, VisibleSet, DesigningSet, DimensionSet, DimensionGet, and StateGet. These methods provide functionality typically associated with a screen control such as an Edit field. Their names describe their functionality. For instance, DimensionSet sets the dimensions for an Edit control, based on passed-in Values for starting Location (x, y), Width (dx) and Height (dy). Similarly, VisibleSet specifies whether the control is currently visible in the user interface (at runtime). This is the interface through which the parent window (i.e., container) interacts with the control, for example, creating it, turning its visibility on or off, sets its dimensions, or the like. When the user re-sizes the Edit control during design, the parent window in turn makes one or more DimensionSet calls.")**

while maintaining said first or second content visibility when performing said at least one of shifting, upsizing or removing, if the corresponding first or second display container cell is not removed. **(e.g. Col 16, Ln 15-30"The Edit class implements the IControlServer methods 1020: ClientSet, ParentWindow, CreatedSet, VisibleSet, DesigningSet, DimensionSet, DimensionGet, and StateGet. These methods provide functionality typically associated with a screen control such as an Edit field. Their names describe their functionality. For instance, DimensionSet sets the dimensions for an Edit control, based on passed-in Values for starting**

Location (x, y), Width (dx) and Height (dy). Similarly, VisibleSet specifies whether the control is currently visible in the user interface (at runtime). This is the interface through which the parent window (i.e., container) interacts with the control, for example, creating it, turning its visibility on or off, sets its dimensions, or the like. When the user re-sizes the Edit control during design, the parent window in turn makes one or more DimensionSet calls." [inherently, for example, turning of the visibility of one component would not affect the visibility of the other])

Response to Arguments

3. Applicant's arguments filed August 25, 2008 have been fully considered but they are not persuasive.

In Remarks, Applicant argues;

More specifically, the Examiner quotes at length various portions of Williams as disclosing the recitations of claim 1. Those portions describe a visual development system which enables a user to graphically build a program. The system is shown in Figure 5 as comprising a plurality of windows, some of the

windows nested within each other. Each of the windows may be a associated with a component (see, e.g., window 550, which is associated with VisualComponentl). Also, components are described as being nestable within each other to an arbitrary depth (Abstract). As shown in Figure 6A, this nesting relationship may be displayed to the user by placing an icon for a child component within a window of a parent component (see,

e.g., icon 620 for VisualComponent2 within the window 630 for VisualComponentl). When the user interacts with the icon, a window corresponding to the component represented by the icon may be opened and displayed to the user (see window 640 in Figure 6A). The Examiner also points to Figure 6C, equating icon 651 to the first display action cell and property menu 652 as the third/fourth display container cell, which is displayed in response to user interaction with icon 651. In response, Applicant has amended claim 1 to recite "the first action being conditioned at least in part on one or more state values or policy attributes of one or more of the display container cells." In contrast, Williams does not disclose the conditioning of actions associated with icons, much less conditioning those actions on state values or policy attributes of the display container cells. The icons of Williams are not described in any great detail and thus are merely state of the art icons which automatically perform an action in response to user interaction.

Examiner's Response:

Examiner respectfully disagrees. With regards to the amended "state values or policy attributes" examiner points to the Property Values described in Col. 11 of Williams that are 'at least in part' conditional to the actions.

The rest of Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to MATTHEW J. BROPHY whose telephone number is 571-270-1642. The examiner can normally be reached on Monday-Thursday 8:00AM-5:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MJB

3/13/2007

/Wei Zhen/

Supervisory Patent Examiner, Art Unit 2191